Replacement Midterm/Final #1 in 2009

Otázky označené $\mathscr{M}$ patria do midtermu, otázky označené $\mathscr{F}$ do finalu.

1 Formalizmus [$\mathscr{M}$10/$\mathscr{F}$15 bodov]

V tejto úlohe kladťte hlavný dôraz na presnosť a korektnosť práce s formalizmom z prednášky.

Bez dôkazu smiete použiť, že funkcie $p(x,y) = x + y$, $k(x,y) = xy$ a $m(x,y) = \max(0, x - y)$ sú primitívne rekurzívne.

a) $\mathscr{M}\mathscr{F}$ (4 body) Dokážte, že funkcie $a() = 3$, $b(x) = 3$ a $c(x,y) = 3$ sú primitívne rekurzívne.

b) $\mathscr{M}\mathscr{F}$ (6 bodov) Dokážte: Ak $f(x)$ je primitívne rekurzívna, tak aj $g(x,y) = \sum_{x \le i < y} f(i)$ je primitívne rekurzívna.

c) $\mathscr{F}$ (5 bodov) Dokážte: Ku každej rekurzívnej funkcii f existuje rekurzívna funkcia g taká, že $\lim_{n\to\infty} \frac{g(n)}{f(n)+1} = \infty$.

2 Riešenie: Formalizmus

a) Nulárnu nulu $z() = 0$ máme medzi základnými funkciami.

Unárnu nulu zostrojíme pomocou primitívnej rekurzie z funkcií z (pre vstup 0 vráť 0) a P_1^2 (pre vstup $n+1$ vráť to isté čo pre vstup n), t. j. $j = PR[z, P_1^2]$.

Binárnu nulu dostaneme napr. ako $d = Comp[j, P_1^2]$.

Hľadané funkcie a, b, c teraz dostaneme trojnásobnou kompozíciou successora s funkciami z, j, d. Napr. teda $a = Comp[s, Comp[s, Comp[s, z]]]$.

b) Použijeme primitívnu rekurziu ako for-cyklus, v ktorom budeme sčitovať hodnoty f.

Oplatí sa vyrobiť si pomocnú funkciu $h(x) = \sum_{i<x} f(x)$. Potom platí, že $g(x,y) = h(y) - h(x)$. Netreba to, ide to aj priamo, ale takýto prístup je pohodlnejší – má tú výhodu, že pri vyhodnocovaní h sčitujeme vždy od nuly.

Pomocou „našej ľudskej“ rekurzie vieme funkciu h definovať nasledovne: $h(0) = 0$ a $h(n+1) = h(n) + f(n)$.

Keď toto chceme spraviť formálne, potrebujeme definovať funkciu, ktorá z hodnôt $h(n)$ a n vyrobí hodnotu $h(n+1)$. Takouto funkciou je funkcia $iter(x,y) = p(x, f(y))$, kde p je plus, ktoré už máme k dispozícii podľa zadania.

Takže poďme na vec:

$$\begin{aligned} f2 &= Comp[f, P_2^2] \\ iter &= Comp[p, P_1^2, f2] \\ h &= PR[z, iter] \\ h1 &= Comp[h, P_1^2] \\ h2 &= Comp[h, P_2^2] \\ g &= Comp[m, h2, h1] \end{aligned}$$

(Poznámka: $f2$ je binárna funkcia, ktorá aplikuje f na svoj druhý parameter. Analogicky $h1$ a $h2$.)

c) Výborným príkladom je funkcia $g(n) = n(f(n) + 1)$. Potom bez ohľadu na to, ako vyzerá f, vždy platí:

$$\lim_{n\to\infty} \frac{g(n)}{f(n)+1} = \lim_{n\to\infty} \frac{n(f(n)+1)}{f(n)+1} = \lim_{n\to\infty} n = \infty$$

No a zjavne je funkcia g vždy rekurzívna. Platí totiž $g = Comp[k, P_1^1, Comp[s, f]]$.

(Poznámka: Tu bola veľmi obľúbená chyba robiť konštrukcie, ktoré nefungujú, ak $\lim_{n\to\infty} f(n)$ nie je nekonečno. Napr. $g(n) = nf(n)$ nestačí.)

3 $\mathscr{M}\mathscr{F}$ Exkurzia do Afriky [10 bodov]

V tejto úlohe sa pýtam na niečo, čo som neprednášal, a chcem vidieť, ako si s tým poradíte.

Stojíte na brehu rieky Kongo. Je tu banánová plantáž. Žije tu kolónia opíc, ktoré sa dajú vycvičiť. Máte tu továreň, ktorá vie len tak zo slnečnej energie vyrábať papierové poháriky. A ako MacGyver, ešte máte žuvačku, kancelársku sponku a tabuľku čokolády. Ľubovoľným spôsobom zostrojte pomocou týchto vecí Turing complete model vypočítateľnosti.

4 Riešenie: Exkurzia do Afriky

Jedna z mnohých možností: Zostrojíme Minského registrový stroj s dvoma registrami. Hodnoty v registroch budú počty pohárikov na ľavom a na pravom brehu rieky. Vystačíme si s dvoma opicami.
Opica Alica bude naučená program registrového stroja, ktorý bude mechanicky dokola vykonávať. Respektíve ak pamäti opice nedôverujeme, môže mať program trebárs vyrytý do kôry banánovníka ako vývojový diagram a nalepenou žuvačkou mať označené, ktorá je aktuálna inštrukcia. Potom vždy len prečíta inštrukciu, vykoná ju a prelepí žuvačku.
Inštrukciu INC odsimuluje tak, že si nechá v továrni vyrobiť nový pohárik a prihodí ho na príslušnú kopu. DEC odsimuluje tak, že jeden pohárik z príslušnej kopy hodí do rieky (Greenpeacu na truc). No a implementácia ZERO prebieha metódou kuknem-vidím.
Druhá opica Bob je učiteľ: Keď už s Alicou starnú, spomedzi zvyšku kolónie vyberie dve mláďatá, z ktorých vychová novú Alicu a nového Boba.

Dôležité body:
Alica vykonáva čisto mechanickú činnosť, na ktorú jej dokonca stačí konečne veľa bitov pamäte.
Pohárikov máme potenciálne nekonečne veľa, vieme teda odsimulovať ľubovoľne veľké hodnoty v registroch.
Kolónia opíc na banánovej plantáži môže tiež vydržať žiť potenciálne do nekonečna. A vďaka Bobovi máme zabezpečenú kontinuitu vykonávania programu aj v budúcnosti.

5 $\mathscr{M}\mathscr{F}$ Známa pôda [$\mathscr{M}$10/$\mathscr{F}$15 bodov]

V tejto úlohe sa pýtam na priamo odprednášané veci, ukážte, že im dostatočne rozumiete.

a) $\mathscr{M}$ Dokážte, že sada Wang tiles existuje ku každému kontextovému jazyku a k ničomu inému.

b) $\mathscr{F}$ (5 bodov) Definujte postupnosť množín $\{W_n\}_{n=0}^{\infty}$ tak, aby platilo:
– každá W_n je rekurzívne vyčísliteľná
– pre každú rekurzívne vyčísliteľnú množinu W existuje n také, že $W = W_n$
– predikát $Patri(x, n) = (x \in W_n)$ je čiastočne rekurzívny.

c) $\mathscr{F}$ (5 bodov) Dokážte/vyvráťte: Existuje funkcia, ktorá rastie asymptoticky rýchlejšie ako S a je rekurzívna.
(S je rýchlejšie rastúci brat Busy Beavera – $S(n)$ je max. počet krokov n-stavového DTS ktorý zastane na ε.)

d) $\mathscr{F}$ (5 bodov) Nájdite rekurzívne vyčísliteľnú funkciu, ktorá nemá žiadne rekurzívne zúplnenie, a dokážte to o nej.

6 Riešenie: Známa pôda

a) Ku každej sade Wang tiles vieme zostrojiť LBA, ktorý bude nedeterministicky hádať a overovať správne dláždenie. Pri každom prechode páskou zľava doprava simuluje prikladanie nového riadku dlaždíc pod riadok, ktorého spodné strany sú práve zapísané na páske. Počas prechodu riadkom si v stave pamätá symbol na pravej strane poslednej priloženej dlaždice. Akceptuje, ak sa mu podarí celý riadok naraz vydláždiť správnym spodným symbolom.

   Opačne, ku každej kontextovej gramatike existuje sada Wang tiles, ktorej korektné dláždenia zodpovedajú odvodeniam od konca. Podrobnejší popis tejto konštrukcie je, ak sa dobre pamätám, v skriptách.

b) Jedna z možných definícií: Zoraďme všetky programy počítajúce čiastočné unárne funkcie podľa dĺžky a v rámci dĺžky lexikograficky. Označme P_n program, ktorý je v tomto poradí na n-tom mieste. Nech W_n je množina tých vstupov, pre ktoré P_n zastane.

   (Na prednáške sme mali analogickú definíciu: W_n bol definičný obor čiastočne rekurzívnej funkcie φ_n.)

   Každá W_n je zjavne rekurzívne vyčísliteľná – keď dostaneme x a chceme povedať, či $x \in W_n$, spustíme P_n na vstupe x, a ak skončí, odpovieme áno.

   Každá rekurzívne vyčísliteľná množina je v našej postupnosti. Totiž to, že je rekurzívne vyčísliteľná, z definície znamená, že jej čiastočná charakteristická funkcia $\overline{\chi}$ je čiastočne rekurzívna. Potom ale existuje program, ktorý $\overline{\chi}$ počíta, a tomuto programu zodpovedá práve dotyčná množina.

   No a predikát $Patri(x, n) = (x \in W_n)$ je čiastočne rekurzívny, lebo naše kódovanie programov do čísel je efektívne – teda z čísla n, ktoré dostaneme na vstupe, vieme zostrojiť program P_n a ten následne spustiť na vstupe x.

c) Keby takáto funkcia f existovala, vedeli by sme rozhodovať, či daný DTS na prázdnom vstupe zastane. Toto ale nevieme, preto taká f neexistuje.

Podrobnejšie: Ak existuje takáto f, tak existuje nejaké n_0 také, že $\forall n \geq n_0 : \ f(n) > S(n)$. Potom definujme g nasledovne: pre $n < n_0$ bude $g(n) = f(n_0)$, a pre $n \geq n_0$ bude $g(n) = f(n)$. Funkcia g sa od f líši len na konečne veľa vstupoch, je teda tiež nutne rekurzívna.

Keď teraz na vstupe dostaneme TS A, môžeme spraviť nasledovné kroky: Zistíme si jeho počet stavov n a spočítame $k = g(n)$. Keďže $k > S(n)$, vieme, že ak A nezastane do k krokov, tak už nezastane nikdy. Preto odsimulujeme k krokov A a pozrieme sa.

d) Riešením je napríklad univerzálna funkcia pre čiastočne rekurzívne funkcie. Meta-argument: keby šla zúplniť tá, tak by šla aj každá iná čiastočne rekurzívna funkcia. Korektný dôkaz bol na prednáške.

Krajšia konštrukcia (podľa Elfinky): Uvažujme funkciu k, ktorá pre ľubovoľné n zostrojí DTS číslo n, ten simuluje na prázdnom vstupe, a ak zastaví, tak vráti počet krokov, ktorý spravil.

Zjavne je k rekurzívne vyčísliteľná. Ak by existovalo rekurzívne zúplnenie k, vedeli by sme pomocou neho rozhodovať klasický problém: či daný DTS na prázdnom vstupe zastane. Stačilo by si spočítať k(jeho číslo), odsimulovať toľko krokov a pozrieť sa.

7 $\mathscr{F}$ (True or False) and Justify [20 bodov]

Uveďte, či je tvrdenie pravdivé a nanajvýš tromi vetami svoj názor zdôvodnite.

(Úplne správna odpoveď je za 2.5 boda. Správna odpoveď úplne bez zdôvodnenia je za 0.5 boda. Nesprávna odpoveď, ako aj správna odpoveď s úplne nesprávnym zdôvodnením, sú za 0 bodov.)

Q1 Ak je čiastočná charakt. funkcia množiny A čiastočne rekurzívna, tak jej charakt. funkcia je rekurzívna.

Q2 Ak je v danej logickej teórii predikát $D(x, y) =$„x je korektné tvrdenie a zároveň y je dôkazom tvrdenia x“ rekurzívny, tak je nutne množina dokázateľných tvrdení rekurzívne vyčísliteľná.

Q3 Ak program v úplne všeobecnom Pascale (pracujúcom s neobmedzene veľkými nezápornými celými číslami) obsahuje nenulový počet while-cyklov, tak k nemu existuje ekvivalentný, ktorý ich obsahuje o jeden menej.

Q4 Nech $\mathcal{A}$ a $\mathcal{B}$ sú dve triedy ekvivalencie relácie $\equiv_m$ a nech pre nejaké $A_0 \in \mathcal{A}$ a $B_0 \in \mathcal{B}$ platí $A_0 \leq_m B_0$. Potom $\forall A \in \mathcal{A} : \ \forall B \in \mathcal{B} : \ A \leq_m B$.

Q5 Existuje nekonečne veľa množín, ktoré sú rekurzívne vyčísliteľné, nie sú rekurzívne a nie sú ani m-úplné.

Q6 Totálna funkcia f vzniká minimalizáciou primitívne rekurzívnej funkcie g. Navyše existuje primitívne rekurzívna funkcia h taká, že $\forall \overline{x} : \ f(\overline{x}) < h(\overline{x})$. Potom f musí byť primitívne rekurzívna.

Q7 Nech P je program vyčísľujúci nezáporné reálne číslo x. Potom je rozhodnuteľné či $x = 0$.

Q8 Existuje nekonečná postupnosť množín $A_1, A_2, \ldots$ taká, že všetky A_i sú rekurzívne vyčísliteľné, ale žiadne dve nie sú rekurzívne separovateľné.

8 Riešenie: (True or False) and Justify

Q1 **FALSE.**

Toto je úplná blbosť, keby to platilo, tak je každá rekurzívne vyčísliteľná množina zároveň rekurzívna.

Q2 **TRUE.**

Keď dostaneme nejaké x, môžeme postupne skúšať všetky y. Ak je x dokázateľné, tak v konečnom čase také y nájdeme, inak budeme hľadať do nekonečna.

Iný pohľad na to isté: Paralelne overujeme D pre všetky dvojice (x, y) a vždy, keď nájdeme dvojicu, pre ktorú platí $D(x, y)$, tak dáme x na výstup.

Q3 **TRUE.**

Jediný problém je redukcia počtu while-cyklov z jedného na nula.

Za dva body je odpoveď, že FALSE, lebo túto redukciu vo všeobecnosti nevieme spraviť. Ku každej rekurzívne vyčísliteľnej funkcii existuje program s jedným while-cyklom, ktorý ju počíta, zatiaľ čo programy, ktoré majú len for-cykly, počítajú iba primitívne rekurzívne funkcie.

Všeobecný Pascal však obsahuje aj iné konštrukcie, ktoré nám umožnia simulovať while-cyklus: napr. goto alebo rekurziu.

Q4 **TRUE.**

Triviálne vyplýva z tranzitívnosti $\leq_m$. Keďže $A, A_0 \in \mathcal{A}$, tak $A \leq_m A_0$. Analogicky aj $B_0 \leq_m B$ a spolu s treťou redukciou $A_0 \leq_m B_0$, ktorá existuje podľa zadania, dostávame $A \leq_m B$, q.e.d.

Q5 **TRUE.**

Okrem iného je to jednoduchá množina z prednášky, a tiež každá množina, ktorá sa od nej líši v konečne veľa prvkoch.

Q6 **TRUE.**

Uvedenú minimalizáciu môžeme nahradiť for-cyklom, ktorého horná hranica bude $h(\overline{x})$.

Iný argument: Toto tvrdenie vyplýva z vety o primitívne rekurzívnej časovej zložitosti. Vieme primitívne rekurzívnymi funkciami t_g, t_h zhora odhadnúť časovú zložitosť g aj h (lebo sú primitívne rekurzívne) a pomocou t_g a t_h vieme zostrojiť primitívne rekurzívnu funkciu, ktorá zhora odhadne čas výpočtu f.

Q7 **FALSE.**

K ľubovoľnému DTS A vieme zostrojiť program P_A, ktorý robí nasledovné: Zo vstupu načíta číslo n, odsimuluje n krokov A na prázdnom vstupe a vráti 0, ak A ešte nezastal a 1, ak už zastal. Takýto program (podľa klasickej Turingovej definície, rozmyslite si, ako ho upraviť pre modernú definíciu) vyčísľuje buď číslo 0, ak A nikdy nezastane, alebo číslo $1/(9 \cdot 10^s)$ ak A zastane po s krokoch.

Ak by sme vedeli rozhodovať problém zo zadania, vedeli by sme mu podstrčiť takéto programy a tak rozhodnúť halting problem.

Q8 **TRUE.**

Napr. $A_i = \{n \mid f_n(n) = i\}$, kde f_n je naše efektívne číslovanie rekurzívne vyčísliteľných funkcií. Ľahko ukážeme, že sú všetky rekurzívne vyčísliteľné. A keby sme vedeli ľubovoľné dve od seba rekurzívne separovať, vedeli by sme separovať aj dvojicu množín $FALSE$ a $TRUE$ z prednášky, čo je spor.